01-T2 - Minecraft Game Mode

Development Document

CS 4850 – Section 03 – Spring 2025, March 9th

Professor Sharon Perry

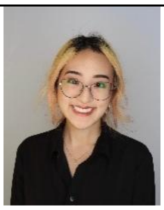
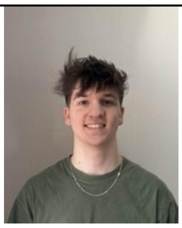
 Ashley Ahn Design	 Matthew Elledge Programmer	AnnaGrace Gwee Design	Bryan Nguyen Team Lead Documentation	Logan Slicker Programmer

Table of Contents

Development Requirements

For future maintainability, the game mode should be updated to GitHub for version control. This has been accomplished by uploading the differing parts of the plugin to branches under the main public repository within an organizational account. Furthermore, the Paper API should be utilized as it is what the KSU Minecraft server staff actively work with. For ease-of-usage, all chat components should also utilize MiniMessage, support full tab completion, permissions, and alias support. These are all accomplishable with the usage of Paper's library and public plugins.

Commands

All commands have their permission nodes defined, and all of the commands are tab-completable alongside being configurable through config and support MiniMessage's formatting. This was completed through Paper and Brigadier.

General

Players are given access to help commands to show them the list of viable commands related to Minecraft BlocksTD. Notably, they will be able to teleport to the hub zone in which they can play the game mode. This has been completed by creating a targetLocation based on the player's coordinates.

Party (Co-Op)

Since the game will support multiple players, players will need to interact with one another. Thus, the party commands come in: Players can invite each other to their parties, accept invites, kick players, warp all members to the hub, and so on. This has been accomplished with a public party plugin.

Gameplay

Players

Players are capable of building and upgrading towers (and thus, managing their economy) and directly interacting with mobs, being able to directly combat them. For mobs, the interaction can remain mostly vanilla, with the key differences being that damage and health is adjusted for balance and that economy should be provided to the player (and split amongst their party if applicable). For towers, players will be provided with a user interface in which they can spend their money to build at designated tower spots, and if there is already a tower, be capable of spending more of their balance to improve the tower's stats.

Mobs

Mobs are to follow the designated paths, provide economy to the players upon death (split between party members if applicable), and should they reach the end of the path, the player will lose lives. These are accomplished through the differing functions specific to their classes: mobHandler, Economy, and RoundManager. As per requirements, these tasks are handled asynchronously, off the main thread.

Towers

Towers should be spawned in their set locations depending on the map/level and will search for mobs traveling along the path. Functions for establishing tower spots, their stats, their costs, and their ability to detect and attack mobs are to be created. Akin to other tower defense games, the towers should be capable of having a queue of targets (closest to furthest) and damaging the mobs with any additional effects to show a hit connecting. These are accomplished within the Tower classes and functions that start when a tower is summoned and end when a tower is destroyed.

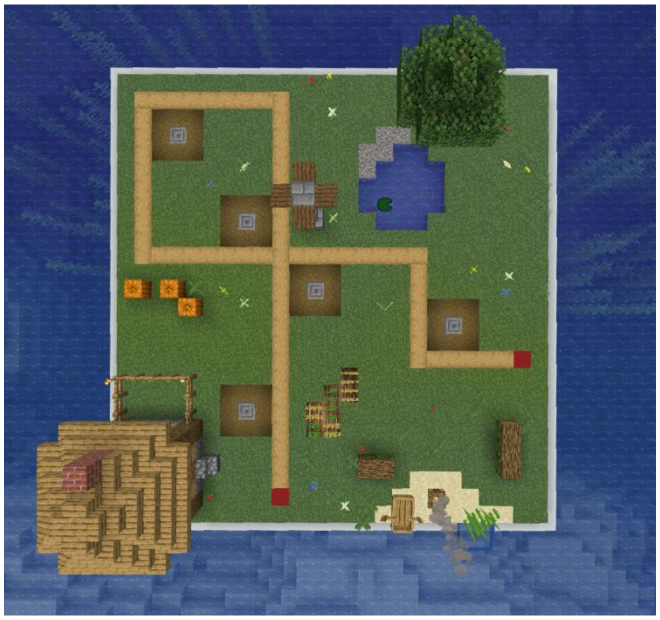
Round Management

Round management will comprise of the bulk of establishing the setting, including the number of rounds, the creation of waves of mobs (and which mobs on which rounds), and the generation of the predefined maps/levels (alongside the locations of buildable tower spots). There is also to be data tracking for player statistics and clear time level leaderboards, and to allow for loading saves upon players leave and their return.

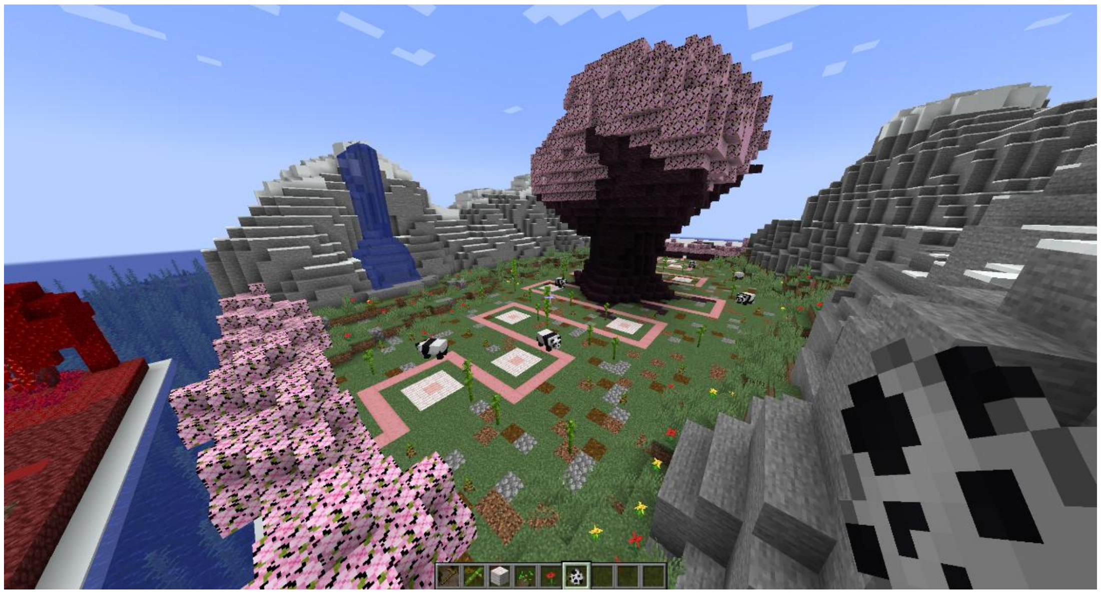
Maps & Levels

Level 1 (Tutorial – Plains)

The chosen biome of inspiration for each level is meant to intuitively reflect its difficulty. Because the tutorial map is meant to be an introduction to the game, the biome chosen was the Plains as it is typically regarded as a "default" biome for Minecraft. The familiarity of this will show players a sense of comfort and safety as they learn to play the game mode. The pathing is the simplest of all maps, with no mountains or visual blocks. The red blocks on each end of the path represent the beginning and end. The stone target blocks with brown dirt surrounding them visually show where the towers will be placed.

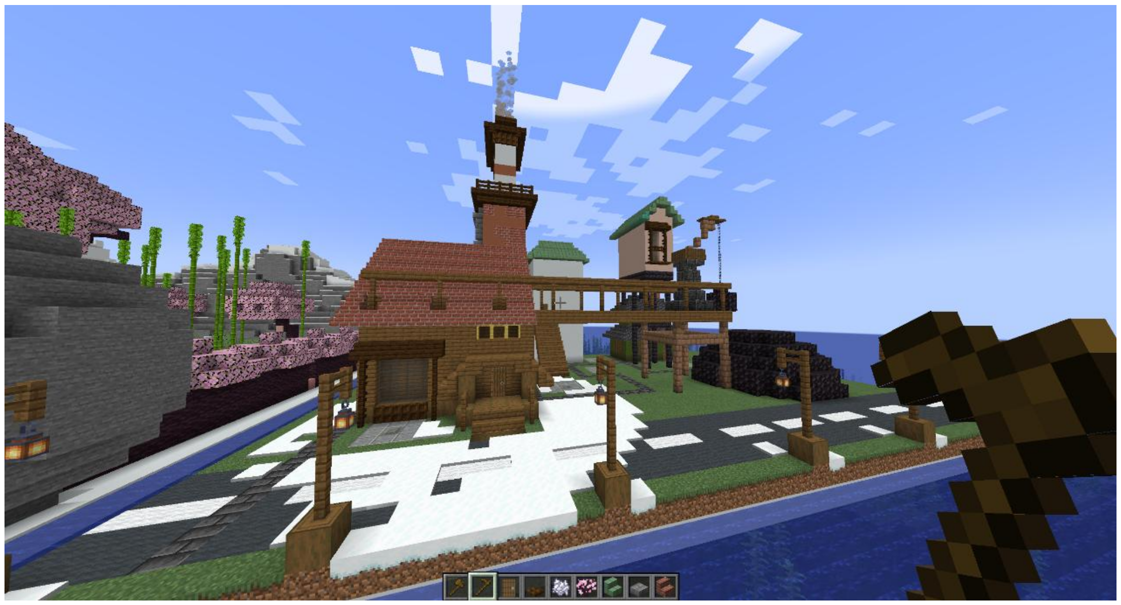


Level 2 (Easy – Cherry Blossom Valley)



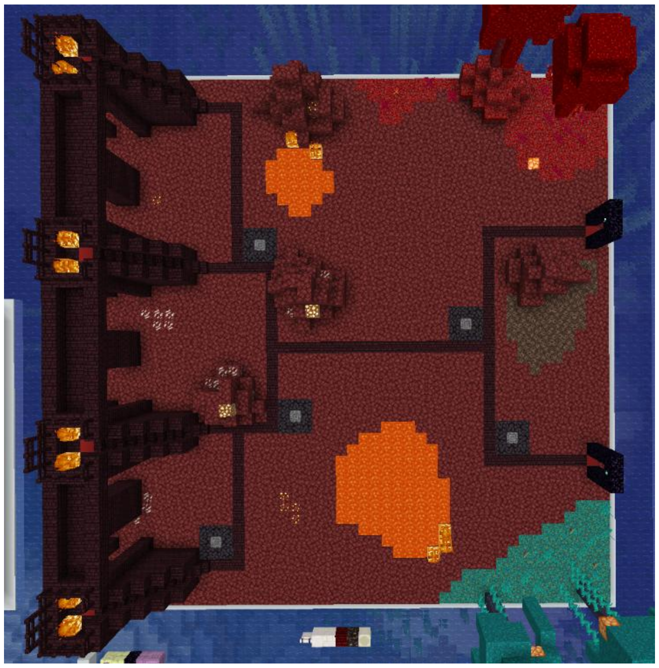
For the second level, the chosen biome is the Cherry Blossom Garden, which provides a visually stunning and serene environment that complements the level's slightly extended length while maintaining an accessible difficulty level. This picturesque landscape is characterized by vibrant pink cherry blossom trees, gently swaying in the breeze, the transition from the foundational tutorial level to more complex challenges. The Cherry Blossom Garden not only offers a more extended gameplay experience but also encourages exploration and collaboration, making it ideal for groups. The paths are designed to meander gracefully through the garden, allowing players to appreciate the scenery while progressively encountering more intricate game mechanics.

Level 3 (Medium – Cityscape)



For the third level, the Cityscape biome introduces an entirely new dimension of complexity and strategy, immersing players in an urban environment bustling with towering skyscrapers, narrow alleyways, and hidden corners. This shift from natural landscapes to an industrious cityscape marks a significant transition in difficulty, presenting unique challenges to seasoned players. The Cityscape is defined by its myriads of tall buildings, each strategically placed to partially obstruct views and create blind spots, demanding heightened awareness and tactical planning from players. Navigating through the intricate maze of urban architecture requires sharp attentiveness, as mobs have numerous opportunities to appear suddenly from behind structures, adding an element of surprise and requiring quick reflexes.

Level 4* (Hard – Nether)



The choice of using the Nether as the environment is to represent how this level will be significantly more difficult to play. The pathing is more complex, and there are visual blockers around the map to limit player vision when attacking mobs or placing towers. The red blocks on each end of the path represent the beginning and end, but in this case, there are now 4 different spots where mobs can emerge from, with 2 points that they are attempting to reach. With this change, players will now have to divert both attention and resources. The grey stone blocks with darker stone surrounding them visually show where the towers will be placed. This might be one of the final levels of the game mode, but it is unconfirmed until all maps are complete.

Database Connection (If Applicable)

There are no database connections planned. This will be updated should this change.

How to Set Project Up (Steps)

Manual Compilation

- Step 1: Compile the plugin (IntelliJ)
  - Open the project in the IDE, select the file option at the top left of the screen, select ‘Project Structure,’ then under ‘Project Settings,’ select ‘Artifacts’
  - Create a new artifact with the plus sign (+) at the top-left of the window, selecting it as a JAR file, also ‘From modules with dependencies…,’ and then selecting OK at the bottom right of the pop-up
  - Change the output directory of the artifact build so it targets the server’s plugins folder, and then click OK

- Step 2: Build the plugin into the server
  - Once the correct build location is in place, again, go to the top-left of the screen and select the ‘Build’ option, and select ‘Build Artifacts…’
  - In the middle of the screen, a pop-up will display, showing the artifact that was created in Step 1. With this, select ‘Build,’ and wait for the building process to complete

- Step 3: Run the server
  - In the server’s directory, it should now contain the desired plugin in the plugins folder
  - Then, run the server (through the .bat file or through the server.jar with its GUI). Once the server is online, you may join it with the plugin enabled
    - To ensure the plugin is running, type “/plugins” and it should display into the chat the plugins the server is running, in which it should display the plugin

Alternate Approach (Download as .jar)

- Step 1: Download the plugin as a .jar file

- Step 2: Drop the jar file into the plugins folder
  - With the server’s next start-up, the newly added plugin should be enabled
  - Alternatively, if the server is currently running, the operators can run the command “/reload confirm” to recompile the server and enable the newly added plugin.